UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/851,554	05/08/2001	Stepan Sokolov	SUN1P833/P6212	4023

22434	7590	02/03/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER .
YIGDALL, MICHAEL J

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 02/03/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

	Applicati n No.	Applicant(s)
Office Action Summary	09/851,554	SOKOLOV ET AL.
	Examin r	Art Unit
	Michael J. Yigdall	2122

-- The MAILING DATE of this communication appears on the cov r sheet with th correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) ☒ Responsive to communication(s) filed on _21 September 2004_.

2a) ☒ This action is **FINAL**.   2b) ☐ This action is non-final.

3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) ☒ Claim(s) _1-4 and 6-21_ is/are pending in the application.

     4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5) ☐ Claim(s) _____ is/are allowed.

6) ☒ Claim(s) _1-4 and 6-21_ is/are rejected.

7) ☐ Claim(s) _____ is/are objected to.

8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) ☐ The specification is objected to by the Examiner.

10) ☒ The drawing(s) filed on _21 September 2004_ is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.

     Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

     Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

     a) ☐ All   b) ☐ Some * c) ☐ None of:

       1. ☐ Certified copies of the priority documents have been received.

       2. ☐ Certified copies of the priority documents have been received in Application No. _____.

       3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

     * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _9/21/04_.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1.   Applicant's response and amendment filed on September 21, 2004 has been fully

considered. Claims 1-4 and 6-21 remain pending.

Response to Arguments

2.   Applicant's arguments have been fully considered but they are not persuasive.

3.   Applicant contends that the verifier described by Steele does not teach determining

whether a Java command is likely to place the only reference to a Java object on the execution

stack (Applicant's remarks, page 10, first paragraph).

        However, as presented below, Steele discloses determining whether stack operands and

the results of instructions or commands are references to objects (see, for example, column 8,

lines 28-39), and likewise discloses determining the number of references (see, for example,

column 16, lines 48-49). Specifically, by determining the number of references to an object,

Steele determines whether there is only one reference to the object.

4.   Applicant contends that Agesen does not teach or suggest determining whether there is a

change in the flow control when it is determined that a command is likely to place a reference to

an object on an execution stack and translating the command into another command when it is

determined that there is a change the flow control (Applicant's remarks, page 10, last paragraph).

        However, as presented below, Steele discloses determining whether a command is likely

to place a reference to an object on an execution stack, due to the invocation of a method or

subroutine (see, for example, column 8, lines 28-39 and column 6, lines 23-36). The invocation

of a method or subroutine is a change in the flow control. Steele also discloses translating the

command into another command (see, for example, column 8, lines 28-39 and column 7, lines 57-64). Furthermore, Agesen discloses determining the control paths in an instruction sequence when there are instructions likely to reference an object (see, for example, column 12, lines 51-67). By determining the control paths in an instruction sequence, Agesen determines whether there is a change in the flow control.

Therefore, in combination, Steele and Agesen teach determining whether there is a change in the flow control when it is determined that a command is likely to place a reference to an object on an execution stack and translating the command into another command when it is determined that there is a change the flow control.

5.   Regarding claim 6, Applicant contends that Steele does not teach determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before the reference is used (Applicant's remarks, page 11, second paragraph). Similarly, Applicant contends that Steele does not teach or suggest determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before the reference is used when it is determined that there is not a change in the flow control and translating the command into another command when it is determined that there is a Putfield command likely to overwrite a reference to an object on the execution stack before the reference is used (third paragraph).

However, the previous Office action did not assert that Steele teaches determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before the reference is used. As presented below, the rejection of claim 6 is based on a combination of Steele and Agesen.

One cannot show nonobviousness by attacking references individually where the

rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ

871 (CCPA 1981) and *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

Information Disclosure Statement

6.   The information disclosure statement filed on September 21, 2004 fails to comply with

the provisions of 37 CFR 1.98 because the copies of the forms from Application No. 09/851,663

do not include the correct application number on each page (i.e., 09/851,544) and do not provide

a blank space next to each citation for this examiner to initial. It is noted that the documents

listed in the copied PTO-1449 forms are duplicates of the documents already considered in the

present application, and likewise at least one document listed in the copied PTO-892 form has

already been cited by this examiner. Accordingly, the information disclosure statement has been

placed in the application file, but the information referred to therein has not been considered.

Drawings

7.   The objection to the drawings is withdrawn in view of Applicant's replacement drawing

sheets filed on September 21, 2004.

Specification

8.   The objection to the attempt to incorporate subject matter into this application by

reference is withdrawn in view of Applicant's amendments to the specification.

Double Patenting

9.   The provisional rejection of claims 1-21 under the judicially created doctrine of

obviousness-type double patenting is withdrawn in view of Applicant's amendments to the

claims.

Claim Rejections - 35 USC § 102

10.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the

basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11.   Claims 9, 10, 15 and 16 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S.

Pat. No. 5,903,899 to Steele, Jr. (art of record, "Steele").

With respect to claim 9 (original), Steele discloses a method of tracking references to

Java objects in a Java programming environment (see, for example, the abstract and column 4,

lines 62-66), said method comprising:

(a) determining whether said Java command is likely to place the only reference to a Java

object on the execution stack (see, for example, column 8, lines 28-39, which shows determining

whether the stack operands and results of an instruction or command are references, i.e.

references to objects; also see, for example, column 2, lines 6-17, which shows identifying

reachable objects, i.e. determining if there are references to an object, and column 16, lines 48-

49, which further shows determining the number of references);

(b) translating said command into another command when said determining determines that said Java command is likely to place the only reference to a Java object on the execution stack (see, for example, column 8, lines 28-39, which shows replacing or translating the instruction into another instruction, and column 7, lines 57-64);

(c) executing said Java command (note that the instructions or commands are inherently executed; also see, for example, column 2, lines 6-17, which shows operating with an executing program);

(d) placing a reference to said object on a reference stack associated with said execution stack when said another command is executed (see, for example, column 7, lines 33-56, which shows placing references on a reference stack during execution); and

(e) wherein said determining of whether said Java command is likely to place the only reference to a Java object on the execution stack is performed during Java Bytecode verification (see, for example, column 16, lines 10-18, which shows performing the determination when verifying Java bytecode).

With respect to claim 10 (original), Steele also discloses the limitation wherein said determining that said Java command is likely to place the only reference to a Java object on the execution stack further comprises determining whether a Getfield, Aload, Getstatic, or Areturn command is being performed (see, for example, column 13, lines 46-56, which shows identifying getfield and getstatic instructions for use with the reference stack, and column 7, line 65 to column 8, line 15, which shows the same for the aload instruction; also see, for example, column 8, lines 28-39, which shows identifying stack results, i.e. return values associated with an areturn command, as references).

With respect to claim 15 (original), Steele discloses a Java Bytecode verifier suitable for

operating in a Java operating environment (see, for example, the abstract and column 4, lines 62-

66; also see, for example, column 16, lines 10-18, which shows a Java bytecode verifier),

(a) wherein said Bytecode verifier operates to determine whether there is at least one Java

command in a stream of Java Bytecode commands such that said at least one Java command is

likely to place the only reference to a Java object on the execution stack (see, for example,

column 8, lines 28-39, which shows determining whether the stack operands and results of an

instruction or command are references, i.e. references to objects; also see, for example, column

2, lines 6-17, which shows identifying reachable objects, i.e. determining if there are references

to an object, and column 16, lines 48-49, which further shows determining the number of

references);

(b) wherein said Bytecode verifier operates to translate said Java command into another

Java command when said Java command is likely to place the only reference to a Java object on

the execution stack (see, for example, column 8, lines 28-39, which shows replacing or

translating the instruction into another instruction, and column 7, lines 57-64); and

(c) wherein a reference associated with said command is placed on a reference stack as

well as said execution stack when said another command is executed (see, for example, column

7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 16 (original), Steele also discloses the limitation wherein said

Bytecode verifier operates to determine whether a Getfield, Aload, Getstatic, or Areturn

command is being performed (see, for example, column 13, lines 46-56, which shows identifying

getfield and getstatic instructions for use with the reference stack, and column 7, line 65 to

column 8, line 15, which shows the same for the aload instruction; also see, for example, column

8, lines 28-39, which shows identifying stack results, i.e. return values associated with an areturn

command, as references).

Claim Rejections - 35 USC § 103

12.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person
> having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the
> manner in which the invention was made.

13.   Claims 1-4, 6-8, 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable

over Steele in view of U.S. Pat. No. 6,047,125 to Agesen et al. (art of record, "Agesen").

With respect to claim 1 (currently amended), Steele discloses a method of tracking

references to objects of an object oriented programming environment (see, for example, the

abstract), said method comprising:

(a) determining whether a command is likely to place a reference to an object on an

execution stack of said object-oriented programming environment (see, for example, column 8,

lines 28-39, which shows determining whether the stack operands and results of an instruction or

command are references, i.e. references to objects).

Although Steele discloses determining whether a command is likely to place a reference

to an object on an execution stack due to the invocation of a method or subroutine, i.e. a change

in the flow control (see, for example, column 8, lines 28-39 and column 6, lines 23-36), Steele does not expressly disclose:

(b) determining whether there is a change in the flow control when said determining determines that said command is likely to place a reference to an object on an execution stack.

However, Agesen discloses determining the control paths in an instruction sequence when there are instructions likely to reference an object, so that the instructions can be rewritten or translated to eliminate conflicts associated with the control paths, thereby improving garbage collection (see, for example, column 12, lines 51-67). Determining the control paths is analogous to determining whether there is a change in the flow control.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Steele with the feature of determining whether there is a change in the flow control, such as taught by Agesen, so that the command likely to place a reference to an object on an execution stack can be rewritten or translated to eliminate conflicts associated with the control paths, thereby improving garbage collection.

Steele in view of Agesen also discloses:

(c) translating said command into another command when said determining determines that there is a change in the flow control (see, for example, column 8, lines 28-39, which shows replacing or translating the instruction into another instruction, and column 7, lines 57-64); and

(d) placing a reference to said object on a reference stack associated with said execution stack when said another command is executed (see, for example, column 7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 2 (currently amended), Steele also discloses the limitation wherein said object-oriented programming environment is a Java compliant operating environment (see, for example, column 4, lines 62-66).

With respect to claim 3 (original), Steele also discloses the limitation wherein said determining of whether a command is likely to place a reference on said execution stack is performed during Java Bytecode verification (see, for example, column 16, lines 10-18, which shows performing the determination when verifying Java bytecode).

With respect to claim 4 (original), Steele also discloses the limitation wherein said determining of whether a command is likely to place a reference on said execution stack operates to determine whether a Getfield, Aload, Getstatic, or Return command is being performed (see, for example, column 13, lines 46-56, which shows identifying getfield and getstatic instructions for use with the reference stack, and column 7, line 65 to column 8, line 15, which shows the same for the aload instruction; also see, for example, column 8, lines 28-39, which shows identifying stack results, i.e. return values associated with a return command, as references).

With respect to claim 6 (currently amended), although Steele discloses determining whether a putfield instruction is likely to place a reference to an object on the execution stack (see, for example, column 13, lines 46-56), Steele does not expressly disclose:

(a) determining whether a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used when said determining determines that there is not a change in the flow control.

However, Agesen discloses determining whether a variable storing a reference is reused or overwritten before the reference is used, which causes a conflict, so that the instruction using the variable can be changed or translated to eliminate the conflict (see, for example, column 12, lines 9-30), thereby improving garbage collection (see, for example, column 5, lines 16-19).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the system of Steele with the feature of determining whether a command is likely to overwrite a reference to an object before the reference is used, such as taught by Agesen, so that the command can be changed or translated to eliminate the conflict, thereby improving garbage collection. Furthermore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the determining in all cases of such conflicts, even when there is not a change in the flow control.

Steele in view of Agesen also discloses:

(b) translating said command into another command when said determining determines that there is a Putfield command is likely to overwrite a reference to an object on the execution stack before said reference is used (see, for example, column 8, lines 28-39, which shows replacing or translating the instruction into another instruction, and column 7, lines 57-64).

With respect to claim 7 (original), Steele also discloses the limitation wherein said reference stack and said execution stack have the same size (see, for example, column 4, lines 23-29, which shows that the sub-stacks used for distinguishing references occupy the same memory as the single stack, i.e. the execution stack; note that such arrangement enables the reference stack and the execution stack to have the same size).

With respect to claim 8 (original), Steele also discloses the limitation wherein at least one reference to an object is stored in an entry with an offset that is the same as the offset used to store said at least one reference in said execution stack, when said another command is executed (see, for example, column 14, lines 21-46, which shows that the references are stored in slots or entries and that the offsets are numbered such that the ordering of the offsets is the same as the original).

With respect claim 20 (currently amended), Steele discloses a computer readable medium including computer program code for tracking references to objects of an object-oriented programming environment (see, for example, the abstract and column 17, lines 20-57), said computer readable medium comprising:

(a) computer program code for determining whether a command is likely to place a reference to an object on an execution stack of said object-oriented programming environment (see, for example, column 8, lines 28-39, which shows determining whether the stack operands and results of an instruction or command are references, i.e. references to objects).

Although Steele discloses determining whether a command is likely to place a reference to an object on an execution stack due to the invocation of a method or subroutine, i.e. a change in the flow control (see, for example, column 8, lines 28-39 and column 6, lines 23-36), Steele does not expressly disclose:

(b) computer program code for determining whether there is a change in the flow control when said determining determines that said command is likely to place a reference to an object on an execution stack.

However, Agesen discloses determining the control paths in an instruction sequence

when there are instructions likely to reference an object, so that the instructions can be rewritten

or translated to eliminate conflicts associated with the control paths, thereby improving garbage

collection (see, for example, column 12, lines 51-67). Determining the control paths is

analogous to determining whether there is a change in the flow control.

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to supplement the system of Steele with the feature of determining whether there is a

change in the flow control, such as taught by Agesen, so that the command likely to place a

reference to an object on an execution stack can be rewritten or translated to eliminate conflicts

associated with the control paths, thereby improving garbage collection.

Steele in view of Agesen also discloses: .

(c) computer program code for translating said command into another command when

said determining determines that said command is likely to place a reference to an object on an

execution stack of said object-oriented programming environment (see, for example, column 8,

lines 28-39, which shows replacing or translating the instruction into another instruction, and

column 7, lines 57-64); and

(d) computer program code for placing a reference to said object on a reference stack

associated with said execution stack when said another command is executed (see, for example,

column 7, lines 33-56, which shows placing references on a reference stack during execution).

With respect to claim 21 (currently amended), Steele also discloses the limitation wherein

said object-oriented programming environment is a Java compliant operating environment (see,

for example, column 4, lines 62-66).

14.   Claims 11-14 and 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over

Steele, as applied to claims 10 and 16 above, respectively, in view of Agesen.

With respect to claim 11 (currently amended), although Steele discloses determining

whether a command is likely to place a reference to an object on an execution stack due to the

invocation of a method or subroutine, i.e. a change in the flow control (see, for example, column

8, lines 28-39 and column 6, lines 23-36), Steele does not expressly disclose the limitation

wherein said determining that said Java command is likely to place the only reference to a Java

object on the execution stack further comprises determining whether there is a change in the flow

control.

However, Agesen discloses determining the control paths in an instruction sequence

when there are instructions likely to reference an object, so that the instructions can be rewritten

or translated to eliminate conflicts associated with the control paths, thereby improving garbage

collection (see, for example, column 12, lines 51-67). Determining the control paths is

analogous to determining whether there is a change in the flow control.

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to supplement the system of Steele with the feature of determining whether there is a

change in the flow control, such as taught by Agesen, so that the command likely to place a

reference to an object on an execution stack can be rewritten or translated to eliminate conflicts

associated with the control paths, thereby improving garbage collection.

With respect to claim 12 (original), although Steele discloses determining whether a

putfield instruction is likely to place a reference to an object on the execution stack (see, for

example, column 13, lines 46-56), Steele does not expressly disclose the limitation wherein said

determining of whether said Java command is likely to place the only reference to a Java object

on the execution stack further comprises determining whether a Putfield command is likely to

overwrite a reference to an object on the execution stack before said reference is used.

However, Agesen discloses determining whether a variable storing a reference is reused

or overwritten before the reference is used, which causes a conflict, so that the instruction using

the variable can be changed or translated to eliminate the conflict (see, for example, column 12,

lines 9-30), thereby improving garbage collection (see, for example, column 5, lines 16-19).

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to supplement the system of Steele with the feature of determining whether a

command is likely to overwrite a reference to an object before the reference is used, such as

taught by Agesen, so that the command can be changed or translated to eliminate the conflict,

thereby improving garbage collection.

With respect to claim 13 (original), Steele also discloses the limitation wherein said

reference stack and said execution stack have the same size (see, for example, column 4, lines

23-29, which shows that the sub-stacks used for distinguishing references occupy the same

memory as the single stack, i.e. the execution stack; note that such arrangement enables the

reference stack and the execution stack to have the same size).

With respect to claim 14 (original), Steele also discloses the limitation wherein at least

one reference to a Java object is stored in an entry with an offset that is the same as the offset

used to store said at least one reference in said execution stack, when said another command is

executed (see, for example, column 14, lines 21-46, which shows that the references are stored in

slots or entries and that the offsets are numbered such that the ordering of the offsets is the same

as the original).

With respect to claim 17 (original), although Steele discloses determining whether a

command is likely to place a reference to an object on an execution stack due to the invocation of

a method or subroutine, i.e. a change in the flow control (see, for example, column 8, lines 28-39

and column 6, lines 23-36), Steele does not expressly disclose the limitation wherein said

Bytecode verifier operates to determine whether there is a change in the flow control.

However, Agesen discloses determining the control paths in an instruction sequence

when there are instructions likely to reference an object, so that the instructions can be rewritten

or translated to eliminate conflicts associated with the control paths, thereby improving garbage

collection (see, for example, column 12, lines 51-67). Determining the control paths is

analogous to determining whether there is a change in the flow control.

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to supplement the system of Steele with the feature of determining whether there is a

change in the flow control, such as taught by Agesen, so that the command likely to place a

reference to an object on an execution stack can be rewritten or translated to eliminate conflicts

associated with the control paths, thereby improving garbage collection.

With respect to claim 18 (original), although Steele discloses determining whether a

putfield instruction is likely to place a reference to an object on the execution stack (see, for

example, column 13, lines 46-56), Steele does not expressly disclose the limitation wherein said

Bytecode verifier operates to determine whether a Putfield command is likely to overwrite a

reference to an object on the execution stack before said reference is used.

However, Agesen discloses determining whether a variable storing a reference is reused

or overwritten before the reference is used, which causes a conflict, so that the instruction using

the variable can be changed or translated to eliminate the conflict (see, for example, column 12,

lines 9-30), thereby improving garbage collection (see, for example, column 5, lines 16-19).

It would have been obvious to one of ordinary skill in the art at the time the invention

was made to supplement the system of Steele with the feature of determining whether a

command is likely to overwrite a reference to an object before the reference is used, such as

taught by Agesen, so that the command can be changed or translated to eliminate the conflict,

thereby improving garbage collection.

With respect to claim 19 (original), the limitations recited in the claim are identical to the

limitations recited in claims 17 and 18 (therefore, see Steele and Agesen as applied to claims 17

and 18 above).

Conclusion

15.   Applicant's amendment necessitated the new ground(s) of rejection presented in this

Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action. In the event a first reply is filed within TWO

MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the date of this

final action.

16.   Any inquiry concerning this communication or earlier communications from the

examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707.

The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the

organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

<div style="text-align:right">

Michael J. Yigdall
Examiner
Art Unit 2122

</div>

MY

mjy

**TUAN DAM
SUPERVISORY PATENT EXAMINER**